CENG4480 Embedded System Development and Applications
The Chinese University of Hong Kong
Laboratory 10: Self-balancing Robot (2) (Software)

Student ID:

2018 Fall

1 Introduction

n this lab you will complete your self-balancing robot by coding the program and tuning the PID constants such that make your robot to standing up. The program flow chart is shown in Figure 1.

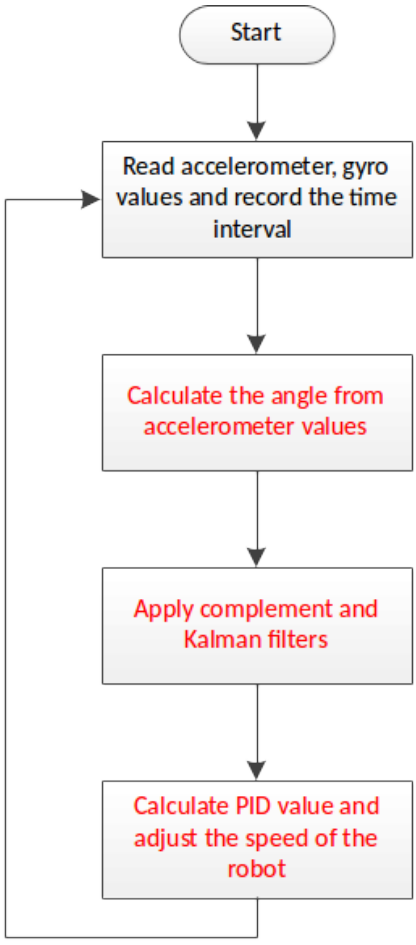

Figure 1: CENG4480 self-balancing robot program flow chart.

2 Objectives

1) To learn how to develop software to control the system.

2) To familiar with the practical work in engineering.

3 Procedures

1) **Calculate the angle from accelerometers values**

On the provided skeleton program Lab10.ino add the angle calculation codes as following:

```
Ayz=atan2(RwAcc[1],RwAcc[2])*180/PI; //angle measured by accelerometer
Ayz-=offset;                         //adjust to correct balance point
```

2) **Add the complement and Kalman filters**

On the provided skeleton program Lab10.ino add the complement and Kalman filters codes as following:

```
Angy = 0.998*(Angy+GyroIN[0]*interval/1000)+0.002*Ayz; //complement
kang = kalmanCalculate(Angy, GyroIN[0],interval);      //kalman
Serial.println(kang);
```

3) **Add the PID calculation and update the speed of motors**

On the provided skeleton program Lab10.ino add the PID calculation and update the speed of motors as following:

```
if ((abs(kang)>=minangle)&&(abs(kang)<maxangle)){
delta = kang;
diff = delta - last;
diff2 = delta - last2;
diff = constrain(diff,-maxdiff,maxdiff);
diff2 = constrain(diff2,-maxdiff,maxdiff);
last2 = last;
last = delta;
LRspeed=P*delta+I*accu*interval*0.001+D*(diff*100+diff2*100)/interval;
accu += delta;
accu = constrain(accu,-maxaccu,maxaccu);
}
else {
LRspeed = 0;
accu = 0;
last=0;
diff=0;
}
```

4) **Calibrate the offset**

- After adding all codes in Lab10.ino then upload it to the Arduino board.
- Hold the robot vertically.
- Open the COM window and find out the offset value.
- Change the offset value in Lab10.ino accordingly.
- Upload the Lab10.ino to Arduino board again.

5) **Tuning the PID constantst**

- Increase the P value in the step of 50 upload to the Arduino each time until the robot start to oscillate (move back and forth).
- Increase I in the step of 50 so that the robot accelerates faster when off balance.
- Increase D in the step of 10 so that the robot would move about its balanced position more gentle, and there shouldn't be any significant overshoots.
- If first attempt doesn't give the satisfying results, reset PID values and start over again with different value of P.
- Repeat the steps until you find a certain PID value which gives the satisfactory results.
- A fine tuning can be done to further increase the performance of PID system.
- In fine tuning, PID values are restricted to neighboring values and effects are observed in practical situations.

6) **Demo your robot to TAs**